软件工程 一班 马启凡 202430551135 **实验名称：有趣的实验（全部已完成）**

**实验内容：**

【1】       猜数游戏。由计算机“想”一个四位数，请人猜这个四位数是多少。人输入四位数字后，计算机首先判断这四位数字中有几位是猜对了，并且在对的数字中又有几位位置也是对的，将结果显示出来，给人以提示，请人再猜，直到人猜出计算机所想的四位数是多少为止。

【2】       自动发牌。一副扑克有52张牌，打桥牌时应将牌分给四个人。请设计一个程序完成自动发牌的工作。要求：黑桃用S(Spaces)表示；红桃用H(Hearts)表示；方块用D(Diamonds)表示；梅花用C(Clubs)表示。按照打桥牌的规定，每人应当有13张牌。在人工发牌时，先进行洗牌……

【3】       在图中的九个点上,空出中间的点,其余的点上任意填入数字1到8;1的位置固定不动,然后移动其余的数字,使1到8顺时针从小到大排列.移动的规律是:只能将数字沿线移向空白的点。请编程显示数字移动过程。



【4】       游戏的目的是用最少的步数将上图中白子和黑子的位置进行交换：游戏的规则是：(1)一次只能移动一个棋子； (2)棋子可以向空格中移动，也可以跳过一个对方的棋子进入空格，但不能向后跳，也不能跳过两个子。请用计算机实现上述游戏。例如：

有三个白子和三个黑子如下图布置：

○ ○ ○ \_ ● ● ●

游戏的目的是用最少的步数将上图中白子和黑子的位置进行交换：

● ● ● \_ ○ ○ ○

黑白子可用不同的符号来表示。

【5】       写一个程序从输入文件中去读取四行大写字母（全都是大写的，每行不超过 100100 个字符），然后用柱状图输出每个字符在输入文件中出现的次数。严格地按照输出样例来安排你的输出格式。（洛谷P1598）

例如：输入

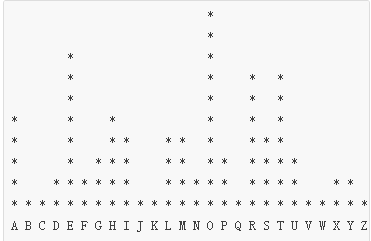
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.

THIS IS AN EXAMPLE TO TEST FOR YOUR

HISTOGRAM PROGRAM.

HELLO!

则输出为：



【1】       猜数游戏。由计算机“想”一个四位数，请人猜这个四位数是多少。人输入四位数字后，计算机首先判断这四位数字中有几位是猜对了，并且在对的数字中又有几位位置也是对的，将结果显示出来，给人以提示，请人再猜，直到人猜出计算机所想的四位数是多少为止。

**1.代码实现**

#include<iostream>

#include<ctime>

#include<cstdlib>

using namespace std;

int main()

{

srand((int)time(0));

int a = rand()% 9999 + 1;

if (a < 1000)

{

a += 1000;

}

cout << a << endl;

int b = 0;

int arra[4];

int c = a;

for (int i = 0; i < 4; i++)

{

arra[3-i] = c % 10;

c = c / 10;

}

while (b != a)

{

cout << "请输入一个四位数字" << endl;

cin >> b;

int d = b;

int arrb[4];

for (int i = 0; i < 4; i++)

{

arrb[3-i] = d% 10;

d = d/ 10;

}

int count = 0;

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

if (arrb[i] == arra[j]&&i==j)

{

cout << "第" << i + 1 << "位数字"<<arrb[i] << "是对的" << endl;

count = j;

break;

}

if(arrb[i] == arra[j] && i != j && j != count)

{

cout << "第" << i + 1 << "位数字"<<arrb[i]<<"的位置不对" << endl;

}

}

}

system("pause");

system("cls");

}

cout << "恭喜您，猜对了！" << endl;

system("pause");

return 0;

}

1. 思路分析

首先通过随机数生成一个四位数，然后将其各位数依次赋值给一个数组，然后让用户输入一个四位数，将其各位数赋值给另一个数组，再依次比较两个数组的每一个元素，并在位置和元素均相同的情况下提示用户，并在位置不对的情况下提示用户，不过，在生成的四位数含有重复数字的情况下，对于位置不同的提示还有特殊情况未考虑。在引入排除变量count后，问题得到了良好的改善。

【2】       自动发牌。一副扑克有52张牌，打桥牌时应将牌分给四个人。请设计一个程序完成自动发牌的工作。要求：黑桃用S(Spaces)表示；红桃用H(Hearts)表示；方块用D(Diamonds)表示；梅花用C(Clubs)表示。按照打桥牌的规定，每人应当有13张牌。在人工发牌时，先进行洗牌……

1. 代码实现

#include<iostream>

#include<cstdlib>

#include<ctime>

using namespace std;

int main()

{

srand((int)time(0));

int arr[52];

for (int i = 1; i <= 52; i++)

{

arr[i - 1] = i;//赋初值

}

int arr1[13];

int S[4] = { 0 }, H[4] = { 0 }, D[4] = { 0 }, C[4] = { 0 };

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 13; j++)

{

int b = rand() % 52;

while (arr[b] == 0)

{

b = rand() % 52;

}

arr1[j] = arr[b];

if (arr1[j] >= 1 && arr1[j] <= 13)

{

S[i]++;

}

else if (arr1[j] > 13 && arr1[j] <= 26)

{

H[i]++;

}

else if (arr1[j] > 26 && arr1[j] <= 39)

{

D[i]++;

}

else

{

C[i]++;

}

arr[b] = 0;

}

}

cout << "A的黑桃数：" << S[0] << " " << "A的红桃数：" << H[0] << endl;

cout << "A的方块数：" << D[0] << " " << "A的梅花数：" << C[0] << endl;

cout << "B的黑桃数：" << S[1] << " " << "B的红桃数：" << H[1] << endl;

cout << "B的方块数：" << D[1] << " " << "B的梅花数：" << C[1] << endl;

cout << "C的黑桃数：" << S[2] << " " << "C的红桃数：" << H[2] << endl;

cout << "C的方块数：" << D[2] << " " << "C的梅花数：" << C[2] << endl;

cout << "D的黑桃数：" << S[3] << " " << "D的红桃数：" << H[3] << endl;

cout << "D的方块数：" << D[3] << " " << "D的梅花数：" << C[3] << endl;

system("pause");

return 0;

}

1. 思路分析

代码在简化后简短很多，主要是因为一开始我为每一个人单独建立了一个数组来记录手上的牌，后来发现不需要，只要每一次用同一个数组进行记录，再用代表黑桃、红桃、方块、梅花的四个数组按照循环次序依次赋值即可。机器发牌不同于人工发牌，实际上并没有物理意义上的“洗牌”，而是不断的进行随机数分配，同时还要保证每一个数都是不同的即可！

【3】       在图中的九个点上,空出中间的点,其余的点上任意填入数字1到8;1的位置固定不动,然后移动其余的数字,使1到8顺时针从小到大排列.移动的规律是:只能将数字沿线移向空白的点。请编程显示数字移动过程。



1. 代码展示

#include<iostream>

#include<cstdlib>

#include<ctime>

using namespace std;

void Graph(char rect[9]);

int main()//main函数

{

int i, j, k;

char Cyc[9] = { 0 };//Cyc[0]到Cyc[7]用于保存8个数，Cyc[8]为空白区

//注意，Cyc是一个字符型数组，现在存入的都只是ASCII码

int CurNode = 9, Temp;

srand((unsigned int)time(NULL));

for (i = 0; i < 8; i++)//随机产生序列

{

Cyc[i] = rand() % 8+ 1; //在前8个位置存入1~8（ASCII代表的数值）

if (CurNode == 9 && Cyc[i] == 1)

CurNode = i;//记录数字1的位置

for (j = 0; j < 8; j++)

{

if (Cyc[i] == Cyc[j] && i != j)

{

i--;

break;

}

}

}

cout<<"原始状态:";

cout << endl;

Graph(Cyc);//调用Graph函数，来展示初始状态的Cyc数组（以 3 \* 3 的形式）

cout<<"开始移动:";

cout << endl;

for (i = 2; i <= 8; i++)//下一个要找的数，经过计算，最多只要操作7次

{

for (j = 0; j < 8; j++)//扫描找数，相当于遍历数组（除了最后一个）

{

if (Cyc[j] == i)//找到了下一个数所在的位置

{

Temp = (CurNode + 1) % 8;//意思就是，按照“顺时针”排序，此时的Cyc[temp]应该为i

if (i != Cyc[Temp])//如果位置正确就不用再移，此时位置不正确，进行移动

{

Cyc[8] = i;//移入中心区（空格区）

Cyc[j] = 0; //移入中心区（空格区）

Graph(Cyc);//向用户展示此时的布局（重新出现空格）

//在已经确定位置不对的情况下，temp（本应该在的地方）和j（实际在的地方）肯定不一样

//下面就进行一次顺时针轮转

if (Temp > j) //如果不小心j的下标小于temp，我们将j加上8，使其始终在temp之前（相当于绕了一圈）

j += 8;

for (k = j; k > Temp; k--)

{

Cyc[k % 8] = Cyc[(k - 1) % 8];//顺时针轮转(只需要在temp和j之间的位置即可)

}

Cyc[Temp] = 0;

Graph(Cyc);//展示布局

Cyc[Temp] = Cyc[8];//从中心区移出

Cyc[8] = 0; //从中心区移出

Graph(Cyc);//展示布局（重新出现空格）

}

CurNode = Temp;//记录当前位置，就是说，此时把排好序的最后一个数的位置作为新的原点，进行下一次排序

break;

}

}

}

cout << "移动完成!";

cout << endl;

system("pause");

return 1;

}

void Graph(char rect[9])

{

int i; char buf[9];

for(i=0;i<9;i++)

{

if(rect[i]) //如果这个位置的元素数值不是0（将字符型进行强制类型转换）

buf[i]=rect[i]+48; //将每个位置的ASCII码加上48，真正成为1~8数字的字符

else buf[i]=' '; //将唯一的一个编码为0的元素赋为空格字符

}

cout << buf[0]<<"-" << buf[1] << "-" << buf[2];

cout << endl;

cout << buf[7]<< "-"<< buf[8] << "-" << buf[3];

cout << endl;

cout << buf[6]<< "-"<< buf[5] << "-" << buf[4];

cout << endl;

cout << endl;

}

1. 思路分析

这道题的关键在于掌握轮换的技巧，并且将一个3\*3的矩阵转化为一个一维数组。首先，如果想要将空格展示出来，要将数组建立为字符型，然后将矩阵正中间的元素初始化为空格，将其余部分用随机赋予的ASCII码值转化为1~8这8个数字。通过一系列演算，可以确定整个过程不算复杂，采用类似于迭代的方式进行循环排序即可。

【4】       游戏的目的是用最少的步数将上图中白子和黑子的位置进行交换：游戏的规则是：(1)一次只能移动一个棋子； (2)棋子可以向空格中移动，也可以跳过一个对方的棋子进入空格，但不能向后跳，也不能跳过两个子。请用计算机实现上述游戏。例如：

有三个白子和三个黑子如下图布置：

○ ○ ○ \_ ● ● ●

游戏的目的是用最少的步数将上图中白子和黑子的位置进行交换：

● ● ● \_ ○ ○ ○

黑白子可用不同的符号来表示。

1.代码展示

#include<iostream>

using namespace std;

unsigned int \_index = 2;//index=3，3表示起始时"\_"所在位置，000\_111,这里写2就是先让0先移动，00\_0111;

void show(char \*Array) {//用于显示数组

for (int i = 0; i<(7); i++)

cout << Array[i];

cout << endl;

}

bool GameOver(char \*Array)//判断游戏是否结束

{

int i = 0;

for (i = 0; i<3; i++)

{

if (\*(Array + i) == '0')

return false;

}

for (i = 6; i >= 4; i--)

{

if (\*(Array + i) == '1')

return false;

}

if (\*(Array + 3) != '\_')

{

return false;

}

return true;

}

void SwapPos(char \*Array, unsigned &\_index) {

//传引用，为了随时改变本身的参数

// \_??????时

if (\_index == 0)

{

while (1)

//最多执行两次 \_01  或者\_10,

{ //如果\_旁边为1直接将1换位

if (Array[\_index + 1] == '1') {

Array[\_index] = '1';

Array[\_index + 1] = '\_';

\_index = 1;

break;

}

else {//\_01

Array[\_index] = '1';

Array[\_index + 2] = '\_';

\_index = 2;

break;

}

}

show(Array);

}

/\*  ??????\_时 \*/

if (\_index == 6)

{

while (1)//最多执行两次 01\_  或者10\_,

{//如果\_旁边为1直接将1换位

if (Array[\_index - 1] == '0') {

Array[\_index] = '0';

Array[\_index - 1] = '\_';

\_index = 5;

break;

}

else {//\_01

Array[\_index] = '0';

Array[\_index - 2] = '\_';

\_index = 4;

break;

}

}

show(Array);

}

//0\_1情况  此时0想向右移动，而1想向左移动

if (Array[\_index - 1] == '0'&&Array[\_index + 1] == '1')

{

if (\_index<3) {//\_index小于4时0先右移

Array[\_index] = '0';

Array[\_index - 1] = '\_';

\_index--;

}

else {//\_index大于3时1向左边移动

Array[\_index] = '1';

Array[\_index + 1] = '\_';

\_index++;

}

show(Array);

}

// 1\_0时 只有旁边的数可能移动，而且是不可能同时出现相向的

else if (Array[\_index - 1] == '1'&&Array[\_index + 1] == '0')

{

if (\_index<3)

{ //\_index小于3先判断\_的右边（也是为了防止越界）1\_00是不可能出现的，因为小于4，还没有结束，0和0是不能在一起的

Array[\_index + 2] = '\_';

Array[\_index] = '1';

\_index = \_index + 2;

}

else

{ //\_index大于3先判断\_的左边边 由上面推理得只能是01\_0

Array[\_index] = '0';

Array[\_index - 2] = '\_';

\_index = \_index - 2;

}

show(Array);

}

//0\_0

else if (Array[\_index - 1] == '0'&&Array[\_index + 1] == '0')//!!直接写连等‘0’不识别

{

if (Array[\_index + 2] == '1') {//0\_01

Array[\_index] = '1';

Array[\_index + 2] = '\_';

\_index = \_index + 2;

}

else {//0\_00 时，左边的0右移

Array[\_index] = '0';

Array[\_index - 1] = '\_';

\_index = \_index - 1;

}

show(Array);

}

//1\_1

else if (Array[\_index - 1] == '1'&&Array[\_index + 1] == '1')

{

if ( \_index>=2&&Array[\_index - 2] == '0') {//01\_1

Array[\_index] = '0';

Array[\_index - 2] = '\_';

\_index = \_index - 2;

}

else {//11\_1

Array[\_index] = '1';

Array[\_index + 1] = '\_';

\_index = \_index + 1;

}

show(Array);

}

}

int main()

{

//展示题目条件

cout << '0' << '0' << '0' << '\_' << '1' << '1' << '1' << endl;

/\*数组初始化\*/

char Array[7] = { '0','0','\_','0','1','1','1' };

/\*用于显示初始化的数组\*/

show(Array);

/\*移动棋子\*/

while (!GameOver(Array))

{

SwapPos(Array, \_index);

}

show(Array);

system("pause");

return 0;

}

2.运行效果

000\_111

00\_0111

0010\_11

00101\_1

001\_101

0\_10101

\_010101

10\_0101

1010\_01

101010\_

10101\_0

101\_100

1\_10100

11\_0100

1110\_00

111\_000

111\_000

请按任意键继续. . .

3.思路分析

这道题可以采用递归或迭代的方式解决，我采用了迭代的方式，主要问题是要控制边界，也就是我们不能让下标有超出数组边界的可能。如果直接看黑白子的移动比较抽象，我们可以换一种思路，考虑空格的移动，在不同情况下判断空格两侧棋子的移动。为了防止棋子的后退，我们要避免“堵塞”现象的出现，也就是有一方的棋子不能继续前进，做到这些，再在每一步中将数组的状态展示出来，代码就完成了。

【5】       写一个程序从输入文件中去读取四行大写字母（全都是大写的，每行不超过 100100 个字符），然后用柱状图输出每个字符在输入文件中出现的次数。严格地按照输出样例来安排你的输出格式。（洛谷P1598）

例如：输入

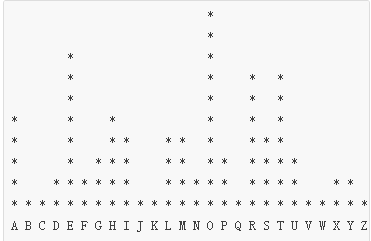
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.

THIS IS AN EXAMPLE TO TEST FOR YOUR

HISTOGRAM PROGRAM.

HELLO!

则输出为：



1. 代码展示

#include<iostream>

#include<string>

using namespace std;

int main()

{

int Letter[26] = { 0 };

for (int i = 0; i < 4; i++)

{

string a;

getline(cin, a);

for (int j = 0; j < a.size(); j++)

{

if (a[j] >= 65 && a[j]<91)

{

Letter[int(a[j]) - 65]++;

}

}

}

char display[100][100];//定义输出数组并初始化

for (int i = 0; i < 100; i++)

{

for (int j = 0; j < 100; j++)

{

display[i][j] = ' ';

}

}

int j = 0;

for (int i = 0; i < 52; i += 2)

{

display[99][i] = (char)(j + 65);

j++;

}

int k = 0;

for (int i = 0; i < 52; i += 2)

{

for (int j = 98; j >= 99 - Letter[k]; j--)

{

display[j][i] = '\*';

}

k++;

}

int max = 0;

for (int i = 0; i < 26; i++)

{

if (max <= Letter[i])

{

max = Letter[i];

}

}

for (int j=99-max;j<=99;j++)

{

for (int i = 0; i < 52; i++)

{

cout << display[j][i];

}

cout << endl;

}

system("pause");

return 0;

}

1. 思路分析

这道题只要正确地采用字符型数组和string类的输入格式控制，对于读入4行字符并不是一件难事。主要是控制输出的格式比较重要。首先我们创建一个100\*100的二维矩阵，然后严格按照要求的格式一次进行赋值与输出，要注意不能输出空行和空格即可。

**实验总结**

**这些实验兼具难度与挑战性，进一步激发了我对于算法和结构的兴趣，同时也锻炼了我调用各种库函数（algorithm除外）的能力，促进了我对于解决实际问题的理解：真正困难的不是主代码的编写，而是对于编译中各种可能出现的问题的预防与规避，做到这些，才能让代码的稳健性与灵活性更进一步，达到合理解决实际问题的程度。**